

Keivan Karimi

Building a mobile application using the Ionic framework

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Thesis

1 March 2017

Author(s) Title	Keivan Karimi Building a mobile application using the Ionic framework
Number of Pages Date	42 pages 1 March 2017
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software development/Network and communication
Instructor(s)	Anssi Ikonen, Senior Lecturer Peter Hjort, Senior Lecturer
<p>The goal of this thesis was to build a mobile application by using the latest available technologies for mobile hybrid web application development. HTML, CSS, JavaScript, AngularJS and Ionic were used to develop a beneficial and user friendly application that would display the latest news in regard to the fashion industry.</p> <p>The outcome of this project was an application called Shiksho. It has been produced and distributed in a local market application store. The application was downloaded for more than 50 times during its first date of publication. The Shiksho application has also received positive feedback from users since the early stages of publication.</p> <p>This application has been developed to fulfil the gap between news websites and mobile applications; the problem that force users to search through different news and media channels to obtain daily news. There is also a possibility for further studies to improve the functionality and interface of the application alongside the data handling and architecture of the application.</p>	
Keywords	application, Ionic, AngularJS, hybrid application

Contents

1	Introduction	1
2	Theoretical background	2
3	Development tools, methods and materials	3
3.1	Operating systems	3
3.1.1	Linux: Ubuntu	3
3.1.2	Windows : Microsoft Windows 7	3
3.2	Development, design and test tools	3
3.2.1	Text editor: Sublime	3
3.2.2	Design: Adobe Photoshop	4
3.2.3	Network monitoring: Wireshark	4
3.3	Version control: Github	4
4	The WordPress and the back-end of the project	5
4.1	Definition of API	5
4.2	Definition of JSON	5
4.3	Definition of REST	6
4.3.1	Architecture of REST API	6
4.3.2	Handling data in REST API	7
4.3.3	Using REST API in WordPress	8
4.4	Using HTTP API request	9
5	The ionic framework	10
5.1	Introducing the Ionic framework	10
5.2	Ionic framework for building mobile applications	10
5.3	Cordova and ngCordova for the Ionic	10
5.4	Using the Android emulator or the Genymotion	11
6	The AngularJS framework	13
6.1	Introducing the AngularJS framework	13
6.2	The AngularJS architecture	13
6.3	Installing and using the AngularJS	14
6.4	Organizing the application code	15
6.5	The AngularJS components and directives	17
6.6	Data handling	17
6.6.1	Expressions	18

6.6.2	Filters	18
6.7	Dependency injection	19
6.8	AngularJS services	20
6.8.1	\$http service	21
6.8.2	Factory	22
6.8.3	Routes and configuring routes in the AngularJS	24
7	Plugins for the Ionic framework	26
7.1	Installing and using the ngCordova plugin	26
7.2	Installing and using the ngStorage plugin	27
7.3	Installing and using the PhoneGap social sharing plugin	29
7.4	Installing and using the Ionic-toast plugin	33
8	Network security test	36
8.1	Man-in-the-middle packet sniffing	36
8.2	Installing and using the Wireshark	36
8.3	Building a Wi-Fi hotspot and packet sniffing	38
9	Conclusion	42
	References	43

1 Introduction

Before the time of mobile web application development, mobile applications were limited to apps that were specifically designed and developed for mobile operating systems. In 2011, there was no easy solution for web developers who wanted to develop and build mobile applications. As a result web developers could not directly develop mobile compatible applications. By introducing Apache Cordova, which was known as PhoneGap, web programmers had a chance to build mobile applications with web technologies.

Apache Cordova has made mobile hybrid application development possible with HTML, CSS, JavaScript and all available libraries of JavaScript. Apache Cordova will wrap HTML, CSS and JavaScript in a way that programmers have the possibility to develop for any available operating system. As a result developers could develop hybrid mobile applications for iOS, Android, Windows phone and other major operation systems. Furthermore, developers can use same code and structure with minor changes for different operating systems for mobile phones.

The goal of this project is to build a mobile application using the available web development tools and Apache Cordova as a wrapper. The application would be called Shiksho and it would show data which are fetched from the WordPress website as a back-end. Finally, couple of network security tests will be performed on the application to check the possible security flaws.

2 Theoretical background

With the introduction of the WP-REST-API and availability of the Apache Cordova, there are great opportunities for developing mobile applications that consume the WordPress API. Web developers who develop WordPress websites and web do not need to use dedicated programming languages for each major mobile operating system anymore. Furthermore the cycle of developing and testing an application on different major platforms such as Android, iOS and Windows phone would be shorter.

As a matter of fact, building mobile web applications and hybrid applications based on other providers API is a common approach in mobile development world. Applications such as Amazon Appstore, Evernote and Twitter are all categorized as hybrid applications. Millions of users interact with these applications on daily basis and these applications. Therefore, they are a lot of potentials in this area to develop applications that can consume data which are built with other web services.

Although it seems that the application market are full of hybrid applications, introduction of the WP-REST-API was a game changer. There are a lot of business that have WordPress websites and at some point they are in need of mobile applications for their websites. Frameworks for developing mobile hybrid applications such as Ionic can full fill this need.

The purpose of this project is to build a mobile hybrid application with the latest available technologies for displaying posts and news from a WordPress website. With this method the end users do not need to navigate to main site to read latest news. Users will receive these news by using the Shiksho mobile application.

3 Development tools, methods and materials

3.1 Operating systems

3.1.1 Linux: Ubuntu

Ubuntu, an ancient African word for 'humanity to others', is an operation system and one of favorite Linux distributions. Ubuntu is available for free and the founder is Mark Shuttleworth. The Ubuntu project is based on the Debian release of Linux. The operation system has two types of releases. One is a short-term release, every 6 months, and the other is a long-term release or LTS which is distributed every two years.

Ubuntu was used as the main operating system for the development of Shiksho application in this project and all the coding parts and software development parts were done on this operating system. During the development of the project the LTS version 14.04 was hired.

3.1.2 Windows: Microsoft Windows 7

One of the most widely known versions of Microsoft Window is 7. It was released in 2009 and until now has the highest market share, approximately 50%, in the computer operating system market. The Windows 7 was used mostly for the design parts of the Shiksho project. Designing the logo, images and color pallets were done in this operation system. Also the process of network monitoring and packet analyzing was done in Windows 7.

3.2 Development, design and test tools

3.2.1 Text editor: Sublime

Sublime is a rich text editor for coding and it provides multiple plug-ins for different development languages. It was used as the main development editor in the Shiksho project. AngularJS plugin was also hired for code efficiency in Sublime.

3.2.2 Design: Adobe Photoshop

Photoshop is a multipurpose design tool and it is developed by the Adobe company. It is one the most used tools for designing and graphic editing. By using this software, designers can edit different aspects of an image such as layers and colors. It can be used to design photographs, mobile and web application user interface designs. Adobe Photoshop was used for the design purpose of the Shiksho project. The logo, main images and color pallets were designed in Photoshop. Furthermore, the basic layouts and sketches were designed in this product of the Adobe company.

3.2.3 Network monitoring: Wireshark

Wireshark is one the most advanced network packet analyzer. It is also a protocol analyzer and it helps developers, system administrators and network administrators to debug network-level problems. Wireshark analyses the network traffic and demonstrates the in use packets. Wireshark was used for packet monitoring and packet analyzing in the Shiksho project. The Wireshark software was used on the Windows 7 machine.

3.3 Version control - GitHub

Version control is a system that can be used to track files and their changes in a project. The VCS, version control system, allows developers to revert back to any states of the project and modify or change any files. Nowadays almost every project is controlled by a VCS, and the Shisko application was not an exception.

There are many VCSs available for software development such as ArX, SVN, Mercurial, Git. Each of them offers multiple features. In the development of the Shiksho application, Git was used as the version control. The source code of the application is available under this URL: <https://github.com/keivan85/blueWP>

4 The WordPress and the back-end of the project

Every application and project that interacts with users and display some data to them needs dynamic data. This dynamic data can be provided by a back-end service. It is possible to build a back-end service from ground and use their API end-points or use a readymade service that offers API end-points. WordPress is one of the web services and content managing systems that also offers end point APIs. For the back-end part of the project, WordPress with wp-rest-api plugin was used.

4.1 Definition of API

API, which stands for Application Programming Interface, is a series of tools, protocols and routines that allows to establish a connection. An API defines how different software components can communicate with each other.

As an example, a USB port on a computer can be symbolized as an API. The USB port is supposed to be used as a connector for USB flash drives and similar applicants. Furthermore, it is also possible to connect other devices such as printers and scanners and so on to the computer by using a USB port. In fact, an API helps and simplifies the processes of exchanging data between different parts of software.

As a matter of fact APIs have been in use for a long time. As an example, whenever users login an application with their Facebook or Google+ credentials, they are using API's. [1, p.7; 2]

4.2 Definition of JSON

JSON, JavaScript Object Notation, is a form of data exchange for serializing for example objects and arrays based on JavaScript. It is human/machine-readable and because of that, it is a perfect choice for cross-platform developments. [1, p.8; 4]

4.3 Definition of REST

REST, Representational State Transfer, is one of the many kinds of APIs which uses HTTP, The Hypertext Transfer Protocol, request to get, put, post and delete data, in an application. The majority of web services and websites such as Amazon's S3 Cloud storage, Google, Twitter and Facebook use REST for their API architecture. [1, p.7; 3]

In REST APIs, instead of using services such as SOAP, Simple Object Access Protocol, or XML-RPC, the HTTP protocol is being used. Therefore, the HTTP handles all the four operations of CRUD, which stands for create, read, update and delete. Another positive aspect is REST independency. As a result, servers based on Linux can easily communicate with clients on Windows or vice versa [1, p.8]. APIs based on REST architecture are called RESTful. REST, an architectural style, mostly focuses on simplicity and efficiency.

4.3.1 Architecture of REST API

The architecture of RESTful service can be categorized in the three main components.

- Resources, key components of RESTful, should be identified by URLs and be accessible by using HTTP commands, which are GET, POST, PUT and DELETE through the system.
- Resources can be cached and the protocol is client/server, stateless with layered hierarchy.
- REST services should be able to communicate with non-REST services, and the other way around. [1, p.11; 5]

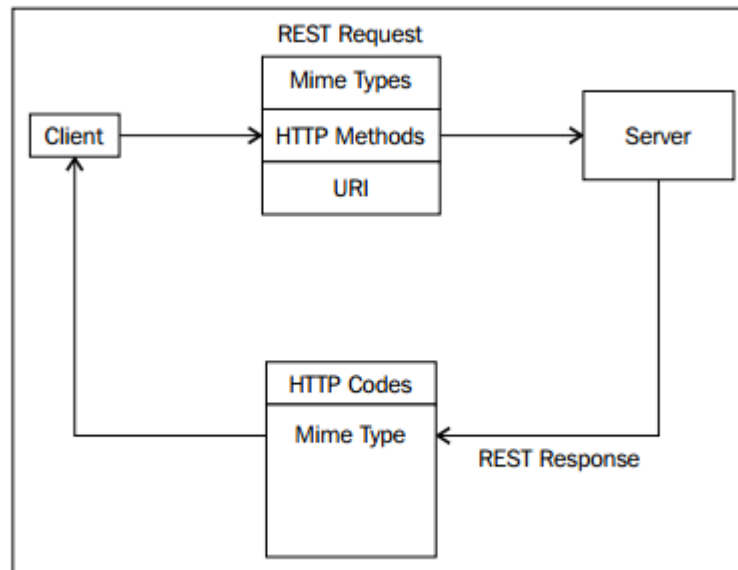


Figure 1. REST architecture components. Copied from [6, p.26]

As illustrated in figure1, the client sends a REST request, which is a standard HTTP methods in MIME, Multipurpose Internet Mail Extensions, and URI, Uniform Resource Identifier, to the hosting server. The server sends back a standard HTTP response. [6, p.26]

4.3.2 Handling data in REST API

For handling data in REST API, depending on the type of request, one of the GET, POST, PUT or DELETE options can be used. As a matter of fact, different HTTP requests can be mapped to each of the CRUD operations.

- The GET request in the HTTP can be assigned to R, which stands for Read in CRUD. As an example, when an application sends a GET request to a web-server for loading contents, it uses Read method.
- The POST, which is another HTTP method, is equal to Create. When an application sends a request to create new resources on a server, it uses the POST method.
- The next method on HTTP request is PUT. The PUT acts similar to Update/Replace in the CRUD method. It can be used to update an original resources on a server. However, it is possible to create resources by using the PUT method.
- The PATCH is another HTTP method. This method is equivalent of Update/Replace on the CRUD methodology. Despite the differences on methodology of a PATCH and a PUT request, in some cases they are being used interchangeably.

- Last but not the least method of the HTTP is Delete. Same method with a same name also exists in the CRUD. Delete is a self-explanatory method and it is used for deleting resources on a server. [24]

For simple queries or for read-only data types, such as reading a blog page, GET is the option. On the other hand, on more complicated queries, such as posting a comment, POST can be used. The main difference is that GET can only read the data while POST can also make changes in the data. [1, p.14]

As an example, the GET request can be used to retrieve data for a user with the first name of John and last name of Doe in this manner: <http://www.allfashion.mobi-proj.com/database/UserDetails?firstName=John&lastName=Doe>.

REST can be used with different programming languages, frameworks and libraries such as Ruby, Java, Python, C#, Angular and React.

4.3.3 Using REST API in WordPress

There are two different ways to use REST API in WordPress:

- Using REST API, which was implemented in the core of WordPress and is available on WordPress version 4.7 and higher.
- Using available REST API plugins for WordPress such as WordPress REST API (Version 2) or JSON API.

The WordPress development team has recently made API access for third-party applications available. This API is in its early stages and the WordPress REST API (Version 2) plugin was used during the development and production of the Shiksho project. Also for sending and receiving a REST request, the Postman app was being used. Postman is a GUI platform that allows to develop and test APIs and their end points [7]. It also allows a user to receive requests as JavaScript, which is a perfect fit for web development [1, p.28].

4.4 Using HTTP API request

With the help of WordPress REST API (Version 2) and Postman application, the data response can be illustrated as in figure 2.

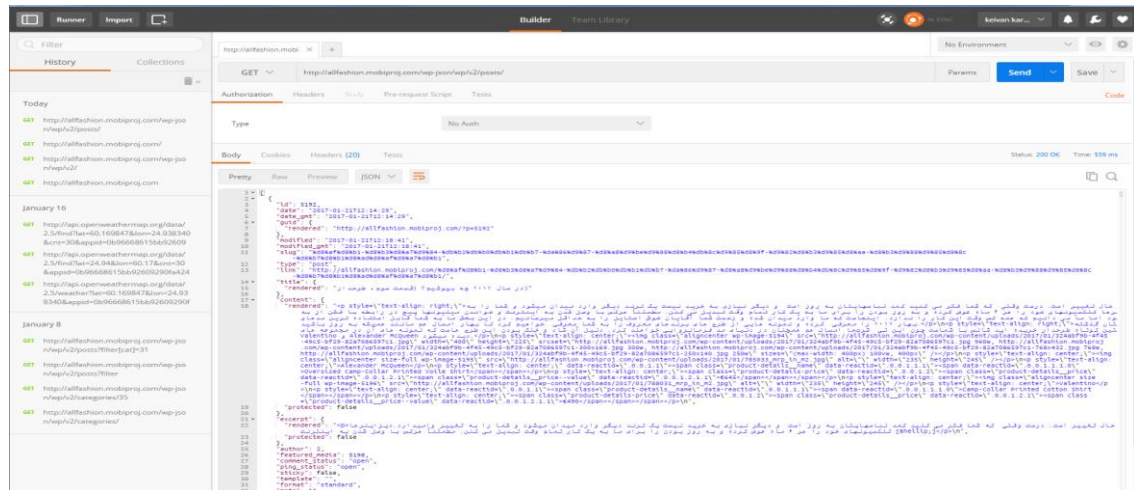


Figure 2. Postman HTTP GET response

As shown in figure 2, different end points of data from a WordPress website can be requested through a HTTP request. For instance, in figure 2, an HTTP GET request was performed to receive the last ten posts, which are available on the website. The request was done on URL: `http://allfashion.mobiproj.com/wp-json/wp/v2/posts/`.

The response is an array of objects, which contains different information. As an example for a single post, the response contains for example id, date, modification date, title and content.

The HTTP request can be performed on different end resources of WordPress REST API (Version 2) plugin. The available resources are posts, post revisions, pages, media, post statuses, comments, taxonomies, categories, tags, users and settings [8].

5 The Ionic framework

5.1 Introducing the Ionic framework

Ionic, which is powered by AngularJS, is one of the latest mobile application development frameworks. Since it is based on AngularJS, it follows the MVW, Model-View-Whatever, or MVVM, Model-View-View-Model architecture [9, 1]. In theory, Ionic is an HTML5 mobile app development framework and it can be used to build a hybrid application. Hybrid applications which are small websites or web services, run in a shell of browser inside mobile operating systems. In this case, it is possible to build applications for iOS, Android and Windows phone with only one time coding. [10]

HTML5 SDK, software development kit, needs native wrappers such as Cordova or PhoneGap and Ionic is not an exception. In fact, by using the Cordova native wrapper an Ionic applications on mobile operation systems will act and will feel similar to a native mobile application. [10]

5.2 The Ionic framework for building mobile applications

As Ionic is based on HTML5, two different approaches can be used for developing applications based on it:

- Developing apps using HTML, JavaScript and CSS.
- Using other methodologies and frameworks such as AngularJS. Something worth mentioning is that most of the core functionality of the Ionic framework is based on AngularJS. Hence, the second approach would be the best choice for web app development. As a result, the AngularJS framework syntax and rules such as directives, controllers and services can be hired in the process of developing Ionic application. [10]

5.3 Cordova and ngCordova for the Ionic

As mentioned in section 5.1, working with Ionic needs a wrapper. Cordova is one of the wrappers. For the needs of the project, an appropriate operating system SDK should be set up on development machine. [9, p.231]

Cordova is suitable for multiple groups:

- Mobile developers, who want to develop for different platforms without re-implementing and re-writing the whole service and application for each platform.
- Web developers, who want to build and deploy web-based applications on different application stores such as iOS app store, Google play and Windows mobile store. [12]

Since the current project was targeted for Android devices, Android SDK should be installed on the development machine.

Android SDK can be downloaded in two ways:

- In Android studio, which is the official IDE for Android, package
- As separate SDK tools [11]

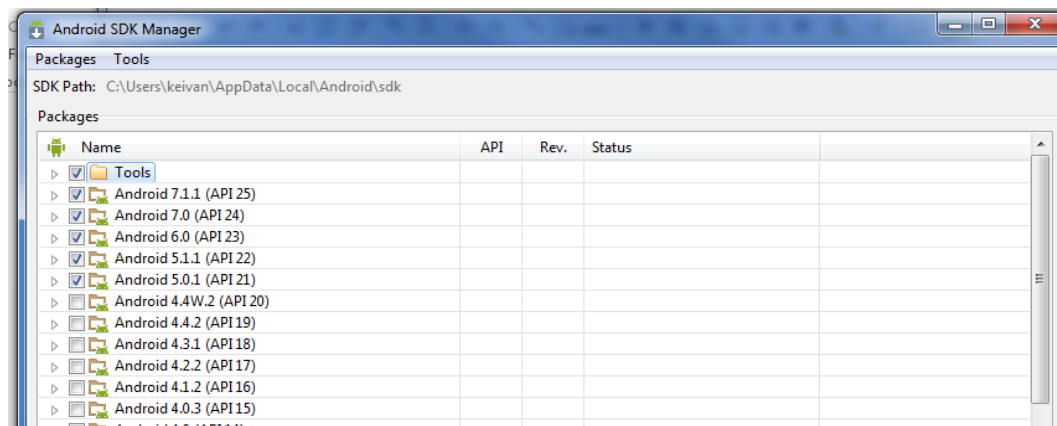


Figure 3. Android SDK manager

As shown in figure 3, after installing Android SDK, Android platform versions and API versions are accessible under the Android SDK manager window. For the best performance of the Ionic framework, Android versions 5, 6 and 7 with API numbers of 21, 22, 23 and 24 were installed and targeted. [11]

5.4 Using the Android emulator or the Genymotion

Every project before deployment needs to be tested. For the testing purpose of Android applications two different tools are available:

- The Android emulator
- The Genymotion

The Genymotion is an Android emulator that can be used to test Android applications on a variety of virtual devices with different specifications. It can be installed on Windows, Linux or Mac OS and it has a wide library of virtual devices.

On the other hand, an Android studio comes with a built-in emulator which can be used for testing purposes. As the Genymotion has better performance and more options over the Android studio native emulator, it was used during the process of development of the project.

6 The AngularJS framework

6.1 Introducing the AngularJS framework

AngularJS is an open-source and client-side JavaScript framework that can be used to develop complex web applications, web services and SPAs, single-page applications. As a matter of fact, AngularJS boosts up plain HTML and adds new syntaxes and structures to it. This boost up leads to new HTML elements and customized attributes. [13, p.1] [14, p.8]

AngularJS is now offered on two versions. The data structure and syntaxes are different in these two versions. While AngularJS version 1 is based on JavaScript ES5 standards, AngularJS version 2 needs Typescript and ES6 standards. For the Shiksho project AngularJS version 1 was hired.

6.2 The AngularJS architecture

AngularJS is inspired by MVC, Model View Controller. In fact, it has MVW, Model View Whatever, design and this design mainly consists of views, models and controllers. The idea behind the 'Whatever' part is that it can be replaced with different methods. Methods such as a controller, which turns MVW to MVC, or Presenter, so that MVW becomes MVP. The presenter pulls data from the model and inject it in to the view. The MVW architecture provides a separation, which is useful for modularity, flexibility and testability, between different layers of application. [13, p.2][13, p.9]

The view, template, is mostly built by HTML or HTML-based languages such as Jade or Pug. The view can benefit two-way data binding, which is one of the most useful features in AngularJS. The controller is a constructor function that enhances the \$scope by simply binding the models and functions to it. [13, p.30]

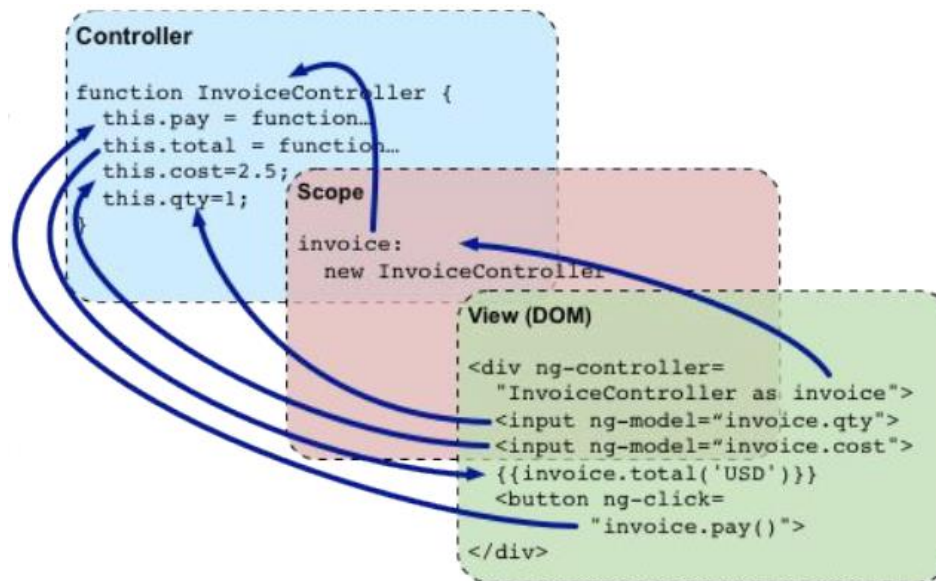


Figure 4. Relation between AngularJS components. Copied from [13, p.10]

According to data in figure 4, a shared object, `$scope`, is the connector or in other words, glue between the view and the model. Using the `$scope` allows developers to exchange information between the model and view [1, p.9]. For example different components of a template can be bound to models and by changing the model the view, or in this case template, will be updated. [13, p.2]

Using AngularJS can benefit developers in multiple ways and one is routing support. Routing support is a major help in SPAs such as Twitter or Chrome app store. By using routing users can navigate to different parts of the application without changing the view. As a result, users have the feeling of working with a native or desktop application. [13, 3]

Another one is templating with HTML. As AngularJS uses basic HTML templating, designers and developers can work side by side. As an example, designers can design a different UI, user Interface, and developers can bind data syntax to different UI components. Every now and then, they can change the UI without the need of making lots of changes to the core of the application. [13, p.3]

6.3 Installing and using the AngularJS

AngularJS can be installed in two ways. One is downloading the zip package from the main website under the address of <https://angularjs.org/>. The second way is installing via

CDN, which stands for Content delivery network. The users can put the link to script in their project and use AngularJS

```
<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/angularjs/1.5/angular.js"></script>
```

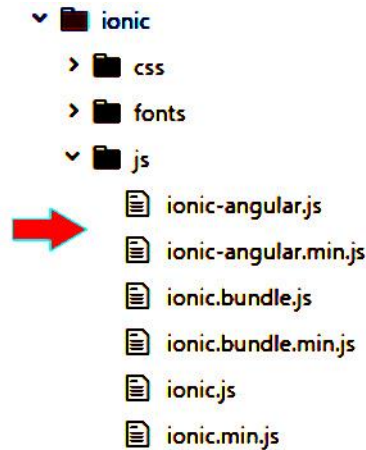


Figure 5. AngularJS in Ionic project

As Ionic has a built-in support for AngularJS and it is constructed on top of AngularJS framework, there is no need to add external AngularJS packages to Ionic apps. Showing in figure 5, the AngularJS is presented in two versions of `ionic-angular.js` and `ionic-angular.min.js` files in the root of the project.

6.4 Organizing the application code

There are multiple approaches to organize an application or code of a project. Some of these approaches are suitable for small projects, while others are the best match for larger projects. The four types of organizing code are illustrated in figure 6.

Domain style

```

app/
  application/
    app.css
    app.js
  login/
    login.css
    loginCtrl.js
    login.html
  parking/
    parking.css
    parkingCtrl.js
    parking.html
  car/
    car.css
    carCtrl.js
    car.html
  lib/
    angular.js
  index.html
  -> files of the application
  -> application module directory
  -> main application stylesheet
  -> main application script
  -> login module directory
  -> login stylesheet
  -> login controller
  -> login view
  -> parking module directory
  -> parking stylesheet
  -> parking controller
  -> parking view
  -> car module directory
  -> car stylesheet
  -> car controller
  -> car view
  -> javascript libraries
  -> AngularJS script
  -> main html file

```

Specific style

```

app/
  css/
    app.css
  js/
    controllers/
      loginCtrl.js
      parkingCtrl.js
      carCtrl.js
    directives/
    filters/
    services/
    app.js
  lib/
    angular.js
  partials/
    login.html
    parking.html
    car.html
  index.html
  -> files of the application
  -> css files
  -> default stylesheet
  -> javascript application components
  -> controllers directory
  -> login controller
  -> parking controller
  -> car controller
  -> directives directory
  -> filters directory
  -> services directory
  -> main application script
  -> javascript libraries
  -> AngularJS script
  -> partial view directory
  -> login view
  -> parking view
  -> car view
  -> main html file

```

Stereotyped style

```

app/
  css/
    app.css
  js/
    app.js
    controllers.js
    directives.js
    filters.js
    services.js
  lib/
    angular.js
  partials/
    login.html
    parking.html
    car.html
  index.html
  -> files of the application
  -> css files
  -> default stylesheet
  -> javascript application components
  -> main application script
  -> all controllers script
  -> all directives script
  -> all filters script
  -> all services script
  -> javascript libraries
  -> AngularJS script
  -> partial view directory
  -> login view
  -> parking view
  -> car view
  -> main html file

```

Inline Style

```

app/
  index.html
  angular.js
  -> files of the application
  -> main html file
  -> AngularJS script

```

Figure 6. Different styles of code for organizing a project (Data gathered from [14, p.13-14-15])

- Inline style is suitable for prototyping or making presentation view of a project or an application. One of the positive points of this style is that it is easy to maintain and build.
- Stereotyped style is the best match for small applications or projects with a limited number of components.
- Specific style is suitable for medium-sized projects. The idea behind specific style is to break components into different parts. As an example developers can break controller.js file into multiple controllers dedicated to specific views.
- Domain style is the perfect choice for big and growing projects. In this style every component and its related view and styling will be nested inside their own directory. [14, p.13-14-15]

After careful consideration for the simplicity of the Shiksho project, the stereotyped styled was hired.

6.5 The AngularJS components and directives

A directive can be defined as an extension for HTML. It can add new syntaxes to HTML and as a result it will extend HTML abilities to create new behaviors [13, p.11] [14, p.18]. Directives can be written in camel case for example ngModel or they can be used with dash or underline, ng-model or ng_model. [14, p.19] However the dash version is more common in the developers' community.

AngularJS like every other frameworks has a couple of built-in directives, which can be hired in different places. For example ngApp, ngModel, ngRepeat or ngBindHtml that has been used in the source code of the Shiksho application in figure 7, can be named.

```

31 <body ng-app="deepBlue" class="deepblue-theme">
32   <ion-nav-view></ion-nav-view>
33 </body>
24   
25     <div class="item-content item-text-wrap text-right p-text-decoration"
26       ng-bind-html="catPost.title.rendered"></div>
27 <ion-list ng-repeat="catPost in category_posts | filter: searchText">
28   <a href="#/app/catPosts/{{catPost.id}}"
29     <div class="item item-image" style="position: relative;" dir="rtl">

```

Figure 7. AngularJS built-in directives

In AngularJS, it is also possible to define custom directives, so that they can be used in different parts of the project. This approach would prevent repeating the same pieces of code multiple times. In the Shiksho project there was no need for using custom directives.

6.6 Data handling

Data handling can be done in AngularJS in multiple ways such as expressions, filters and form validation. AngularJS has a complete support of showing, deploying and transforming, synchronizing, and validating retrieved data on different users' interfaces. [14, p.53]

Expressions and filters were used during the development process of the Shiksho application. Hence, there was no need for users to register for using the application. Form validation was not used at this stage of development.

6.6.1 Expressions

Expression is a part of AngularJS core and it is predefined. It can be hired in view by using curly brackets or binding it directly to the ng-bind-html directive. By using expression developers can interact with attributes bounded to \$scope. [14, 53]

```

7 <ion-content overflow-scroll="true">
8   <div class="item-image" style="position:relative;" dir="rtl" ng-cloak>
9     
10    <p class="item-content item-text-wrap text-right p-text-decoration" ng-bind-html="post_title"></p>
11    <button class="item-font button icon icon-fw-share button-clear" style="color:#0a1612; font-size: 16px !important;"
12    ng-click="share(post_title + ' '
13    + ' 'رو خوندم، خیلی دوستش داشتم، تو هم از اپلیکیشن شیکشو بخونش، از کافه بازار دانلودش کن ')"
14    dir="rtl">کیار نشر
15  </div>
16
17  <ion-card class="item" dir="rtl">
18    <ion-list>
19      <p class="text-right item-text-wrap" style="color:#0e0d0d !important;" ng-bind-html="post_content"></p>
20    </ion-list>
21  </ion-card>
22 </ion-content>

77 function(returnedData){
78   $scope.postDetails = returnedData.data;
79   $scope.post_title = $sce.trustAsHtml($scope.postDetails.title.rendered);
80   $scope.post_content = $sce.trustAsHtml($scope.postDetails.content.rendered);
81   $scope.post_image = $scope.postDetails.better_featured_image.source_url;
82 }

```

Figure 8. Two-way data binding in view and model

As illustrated in part two of figure 8, after receiving data in function on line 77, the data is divided into and assigned to three different variables, which are post_title, post_content and post_image. This is the model part of the application. On the other hand, in part one of figure 8, the returnedData from model is bound to view on lines 9, 10 and 19. This is the view part of the application.

6.6.2 Filters

AngularJS offers multiple filters and they can be used in different ways in the views of an application. It is also possible to define custom filters by calling the function angular.module().filter() and it should be registered in the module of the application. Some of the built-in filters are: currency, date, limitTo, lowercase, json and number. [13, p.233] [14, p.61]

```

11      <i class="icon ion-ios-search placeholder-icon"></i>
12      <input type="search" placeholder="جستجو..." ng-model="searchText" ng-change="searchTextChanged()" />
13    </label>
14  </div>
15
16  <ion-content class="has-header has-subheader" delegate-handle="mainScroll" overflow-scroll="true">
17    <ion-refresher
18      pulling-text="روز رسانی مطالب" on-refresh="doRefresh()">
19    </ion-refresher>
20
21    <ion-list ng-repeat="catPost in category_posts | filter: searchText">
22      181      $scope.searchTextChanged = function() {
23      182        $ionicScrollDelegate.$getByHandle('mainScroll').scrollTop(true);
24      183      };

```

Figure 9. Using a custom filter in Shiksho application

As stated in figure 9, an input search filter is used in Shiksho application. This filter has been used in multiple views.

In figure 9, on line 12 an ng-model directive, searchText, is defined and bound to searchTextChanged() function inside the ng-change directive. The filter is applied to the ng-repeat directive on line 21 to filter out text content on the desired view.

6.7 Dependency injection

Dependency injection or DI in frameworks allows the developer to create reusable components. Furthermore, it clarifies the way components hold their dependencies. Angular DI takes care of creating components, alongside resolving their dependencies. It also makes each component accessible to other components in the case of a request.

There are two ways that components can access their dependencies:

- Invoking a 'new' method on a constructor function. In this way the 'new' method can access the dependency.
- Using global variables to search for dependency.

Both methods have advantages and disadvantages. In the case of using the 'new' method, two components will share the same 'new' method and this will tie them together. Hence, unit testing would be very complicated.

If the second method is hired, a global object will be passed around the application and multiple components will know about it. This method is also not very test-friendly, since

each component should know as little as possible from the other [13, p.226]. The service locator of the AngularJS is used to prevent the first and the second problem.

```

68 // post controller.
69 .controller('postCtrl', function(
70     $scope,
71     $http,
72     $stateParams,
73     $sce,
74     $localStorage,
75     $cordovaSocialSharing) {
76     $http.get('http://allfashion.mobiproj.com/wp-json/wp/v2/posts/' +
77         $stateParams.postId, {cache: true}).then(
78         function(returnedData){
79             $scope.postDetails = returnedData.data;
80             $scope.post_title = $sce.trustAsHtml($scope.postDetails.title.rendered);
81             $scope.post_content = $sce.trustAsHtml($scope.postDetails.content.rendered);
82             $scope.post_image = $scope.postDetails.better_featured_image.source_url;
83
84             }, function(err){
85                 console.log(err);
86             })
87
88     $scope.share = function(title) {
89         //console.log(title);
90         $cordovaSocialSharing.share(title);
91     };
92
93 })

```

Figure 10. Dependency injection to post controller

In figure 10 as an example, dependency methods are injected to the post controller. In fact, AngularJS checks for the registered services and find their matching names and injects them automatically into an argument. In this case the DI system in AngularJS takes care of dependencies.

6.8 AngularJS services

Services are singleton objects that can be used multiple times in different parts of the application such as filters, controllers or directives. Services can be defined in five ways. These five methods are value, constant, service, factory and provider.

- **Module.value:** useable for storing basic values. They can be changed in runtime.
- **Module.constant:** useable for storing basic values. They will not change.
- **Module.service:** the most useable type of services. It is suitable for object-oriented programming and a service can be defined by constructor method.

- `Module.factory`: It is suitable for “Revealing module pattern” and a service can be defined by constructor method.
- `Module.provider`: It is useable for defining and controlling the behavior of a service. [15, p.60-61]

6.8.1 \$http service

AngularJS offers a couple of built-in services and `$http` is one of them. `$http` uses XMLHttpRequest of browser or JSONP with the server via http protocol. `$http` is built with the idea of deferred or promise on `$q` service [3, p.63]. As a web application needs data communication to receive and send data to the back-end, `$http` service is one trustworthy, stable and useful method. [15, p.63]

Promise can be defined as a simple deal. Imagine a scenario of two people checking the weather for fishing. One is staying at home and the other one goes to the local weather forecast office. If the first person comes back with the “sunny” news they will go for fishing and if comes back with “rainy” they will not go. In a case that the first person “does not come back” at all, again they will not go. [16]

The same principles apply to a `$http` request. For instance in figure 11 on line 76, the application sends a `$http` request, and on lines 78 and 84 receives the responses. In the case the response is success (sunny), the data will be retrieved and attached to its appropriate `$scope` variables. If there is a problem in the response or it is not delivered fully (rainy, does not come back), the function on line 84 will be called and it will throw an error in console log regarding the problem.

```

68 // post controller.
69 .controller('postCtrl', function(
70     $scope,
71     $http,
72     $stateParams,
73     $sce,
74     $localStorage,
75     $cordovaSocialSharing) {
76     $http.get('http://allfashion.mobiproj.com/wp-json/wp/v2/posts/' +
77         $stateParams.postId, {cache: true}).then(
78         function(returnedData){
79             $scope.postDetails = returnedData.data;
80             $scope.post_title = $sce.trustAsHtml($scope.postDetails.title.rendered);
81             $scope.post_content = $sce.trustAsHtml($scope.postDetails.content.rendered);
82             $scope.post_image = $scope.postDetails.better_featured_image.source_url;
83
84             }, function(err){
85                 console.log(err);
86             })

```

Figure 11. Promise and \$http service in AngularJS

As The Shiksho application is developed based on data of the WordPress website, the website itself and its API end-points act as the back-end for the Shiksho application.

6.8.2 Factory

AngularJS allows the developer to create a factory function. In theory the factory, which is an injectable type, is a function for creating objects. It can be called on `angular.module()`. [13, p.103-104] [14, p.70]

During the development of the Shiksho application, there was a need for a factory injection, so that it could handle some static data. This part of data needs to be stored on user's device inside application, so it prevents unnecessary network communication and requests.

```

3  angular.module('deepBlue.services', [])
4
5  .factory('BackendService', ['$http', function ($http) {
6
7      var svc = {};
8
9      svc.getProducts = function(){
10         return $http.get('sampledata/products.json');
11     }
12
13     return svc;
14 }])

```

1

```

29 // main controller.
30 .controller('mainCtrl', function($scope,
31     $ionicActionSheet,
32     BackendService,
33     $http,
34     $sce,
35     ionicToast) {
36
37
38     $scope.siteCategories = [];
39
40     $scope.doRefresh = function(){
41         BackendService.getProducts()
42         .success(function(newItems) {

```

3

```

1  [
2
3      {
4         "image" : "sampledata/products/1.png",
5         "id" : 34
6     },
7     {
8         "image" : "sampledata/products/2.png",
9         "id" : 33
10    },

```

2

Figure 12. Using factory services

According to figure 12, in part one there is a factory \$http call for products.json, which is holding an array of objects, file. In part two of figure 12, there are multiple static objects in product.json file and each has two name values. Those two name values are id and path to the image file. Finally in part three, there is a call for getproducts(), which contains the static data, function on line 41 and it binds those data to the main view.



Figure 13. Results of factory service in view of Shiksho application

As illustrated in figure 13, the final result of using factory services is visible. At this point, users' device do not need to make one extra \$http request every time to get the same category images whenever users run and open the Shiksho application.

6.8.3 Routes and configuring routes in the AngularJS

\$routeProvider is a function that allows developers to define and modify the different routes in their application. Simply the routes are used to navigate users to different parts of the application when they click on different URLs. \$routeProvider has four different methods, which are controller, templateUrl, resolve and redirectTo.

- Controller: this method is assigned to the controller related to the view.
- templateUrl: this is the URL that bound to view of the route.
- Resolve: this method is optional and it contains a list of dependencies that should be injected to controller.
- redirectTo: this is the method for redirecting to the location of the view.

There is also “otherwise” function, which is called whenever the route doesn’t match the definition. [14, p.87-88]

```

30  .config(function($stateProvider, $urlRouterProvider) {
31
32
33      $stateProvider
34
93      .state('app.post', {
94          url: '/catPosts/:postId',
95          cache : false,
96          views: {
97              'menuContent': {
98                  templateUrl: 'templates/post.html',
99                  controller : 'postCtrl'
100          }
101      }
102  })
103
104
105  // If none of the above states are matched, use this as the fallback
106  $urlRouterProvider.otherwise('/app/start');
107

```

Figure 14. AngularJS \$routeProvider

As visible in figure 14, on line 30 \$stateProvider and \$routeProvider are used in config service. The URL, which navigates to each posts are defined on line 94. The template

view, which is assigned to this route is defined on line 98 and the controller related to this route is modified on line 99. On line 106 the otherwise method is present and in case routes do not match their definitions it will be called. The application users will be navigated to start page view.

7 Plugins for the Ionic framework

During the development of the Shiksho application, multiple plugins were used for better performance and simplifying the process of development. Some of these plugins are ngCordova, ngStorage, PhoneGap social sharing and Ionic-toast.

7.1 Installing and using the ngCordova plugin

ngCordova, which is made over Cordova API has over 70 different AngularJS extensions to build and deploy Cordova mobile applications with the core of the AngularJS. It gives access to some of the native features of mobile devices such as using camera to take picture or scanning barcodes, finding the current geolocation of device and using device built in flash light. [17]

ngCordova can be installed by using the Bower service or by just simply downloading a zip file and locating its .js files in root of the project.

Using bower:

```
$ bower install ngCordova
```

Using zip file:

Users should include the ng-cordova.js or the ng-cordova-min.js in the index.html of their project. It is important to place the ng-cordova.js/ng-cordova-min.js as shown in figure 11 before cordova.js and after angular.js in index.html file. Otherwise, the dunctions will not be called in correct sequences.

```

11      <!-- ionic/angularjs js -->
12      <script src="lib/ionic/js/ionic.bundle.js"></script>
13      <script src="lib/ngstorage/ngStorage.min.js"></script>
14      <script src="lib/ionic-toast/dist/ionic-toast.bundle.min.js"></script>
15
16      <script src="lib/ngCordova/dist/ng-cordova.js"></script>
17      <script src="cordova.js"></script> <!--chnage to android-->

```

Figure 15. Locating ngCordova in index.html file

As shown in figure 15, on line 12 ionic.bundle.js file, which contains address to angular.js file is placed and on line 16 the ng-cordova.js file is added. Finally on line 17 the main cordova.js file is addressed.

```
3 angular.module('deepBlue', ['ionic', 'ionic-toast', 'deepBlue.controllers', 'deepBlue.services', 'ngStorage', 'ngCordova'])
```

Figure 16. Inject ngCordova in Angular module

After addressing the .js file of ng-cordova, it should be injected as a dependency to Angular module as it is illustrated on figure 16. By adding ngCordova to root of the project, developers can install their desired plugins with the help of Cordova CLI [17]. Developers can add plugins with 'cordova plugin add ...' command

7.2 Installing and using the ngStorage plugin

ngStorage is a module of the AngularJS which has two services: the first one is \$localStorage and the second one is \$sessionStorage. By using this module web storage can be done in the way that AngularJS is implemented.

There are a couple of benefits for hiring ngStorage module.

- Revoking the need of using Getter and Setter: in contrast with other frameworks, simple straight JavaScript can be used and there is no need to wrap the model in accessor methods.
- Better written and implemented code: The module is written according to the AngularJs documentations. As a result it is instructed in a correct way and writing tests for the module is easier.
- Preventing cookies lost or fallback: by using web storage, the number of fallbacks or cookies lost will be decreased significantly.

The module can be installed in multiple ways. One way is to use the NPM, Node Package Management, another way is to use the bower service. There is also possibility to use the nugget, the CDN or the jsDeliver. Since in this project the bower is used as default package manager the module can be installed with this 'Bower install ngstorage' bower command. [18]

ngStorage can be used as an injected module to services. Normally it is injected in the module of AngularJS in the app.js file. The implementation is shown in figure 17.

```

2
3 angular.module('deepBlue', ['ionic', 'ionic-toast', 'deepBlue.controllers', 'deepBlue.services', 'ngStorage', 'ngCordova'])
4
5 .run(function($ionicPlatform, $rootScope, $timeout, $state, $cordovaPush) {
6   $ionicPlatform.ready(function() {

```

Figure 17. Injecting ngStorage in angular module

As shown in figure 17, after the injection of the ngStorage it can be used in controllers and other directives.

Depending on need of using the local storage of device, the ngStorage is useable in different controllers. In case of this project, ngStorage is hired to store the liked posts of the users. By using this module, users can access the list of their favorite posts in the Shiksho application.

```

273
274 .controller('favCtrl', function(
275   $scope,
276   $http,
277   $localStorage,
278   $sce,
279   ionicToast,
280   $cordovaSocialSharing) {
281
282   $scope.doRefresh = function(){
283
284     $scope.Favorites = $localStorage.Favorites;
285     $scope.favorite_posts = [];
286     //console.log($scope.favorite_posts);
287     if($scope.Favorites && $scope.Favorites.length > 0) {
288       $scope.Favorites.forEach(function(element, index, array){
289         $http.get('http://allfashion.mobiproj.com/wp-json/wp/v2/posts/' + element).success(
290           function(data){
291             $scope.favorite_posts.push(data);
292             //console.log(data);
293
294             if($scope.favorite_posts.length == $scope.Favorites.length) {
295               $scope.favorite_posts.forEach(function(element, index, array) {
296                 element.excerpt.rendered = element.excerpt.rendered.substr(0, 150);
297                 element.excerpt.rendered = $sce.trustAsHtml(element.excerpt.rendered);
298                 element.title.rendered = $sce.trustAsHtml(element.title.rendered);
299                 //console.log($scope.favorite_posts);
300                 if($scope.Favorites.indexOf(element.id) != -1) {
301                   element.isFavorite = true;
302                 } else {
303                   element.isFavorite = false;
304                 }
305               })
306             } else {
307               $scope.$broadcast('scroll.refreshComplete');
308             }
309           }
310         })

```

Figure18. Using ngStorage in controller

As shown in figure 18, the ngStorage is used as a dependency called \$localStorage on line 277. On line 284 previous content of \$localStorage.Favorites is put in \$scope.Favorites and on line 285 an empty array of favorite posts is created.

On line 287 the content and the length of Favorites will be checked and if the result is true, the list of posts will be bound to favorites.html view. It is possible to remove a post from the favorite list by using a 'if statement'. On line 300 the indexOf each \$scope.Favorites is checked and in case it is not -1, unselected, the element will be marked as selected. Otherwise it will be marked as unselected.

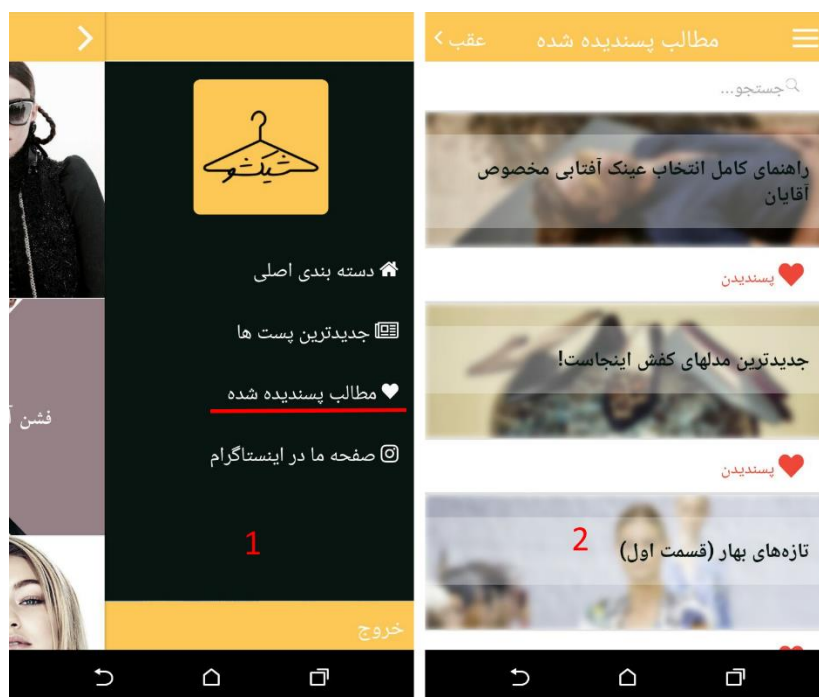


Figure 19. ngStorage result in application view

The final result of using the ngStorage can be seen in figure 19. Here on the left side and part one of the figure, user can open a side menu, which has a list of different navigations. One is a navigation to liked or favorited posts. By touching on that, user will be navigated to the favorite.html which is the template for showing the user's favorite posts. On the right side of figure 19 part two, users can simply remove posts from their favorite lists by touching the red heart icon.

7.3 Installing and using the PhoneGap social sharing plugin

PhoneGap social sharing is one of the best plugins available for adding sharing feature to the phonegap/cordova application. It is meant to work with frameworks that use the phonegap/cordova as a wrapper. Hence, the Ionic and the AngularJS can benefit from

this plugin. The plugin allows users to share a text, a file or a URL using native sharing feature which is available for iOS, Android or Windows phone.

The PhoneGap social sharing plugin is working with the Android version 2.2 or higher, the iOS6 and up and the Windows Phone 8 and up and it has the official support of PhoneGap Build. It also allows users to directly share their files, texts or links on social media applications such as Twitter, Facebook and Instagram.

The plugin can be used in a project in four ways.

First, users can run the

```
$ phonegap local plugin add https://github.com/EddyVerbruggen/SocialSharing-PhoneGap-Plugin.git
in PhoneGap CLI.
```

Second, users can run

```
$ cordova plugin add cordova-plugin-x-socialsharing
$ cordova prepare
in Cordova CLI.
```

Third, the plugin can be installed manually by adding

```
<!-- for iOS -->
<feature name="SocialSharing">
  <param name="ios-package" value="SocialSharing" />
</feature>
<!-- for Android (you will find one in res/xml) -->
<feature name="SocialSharing">
  <param name="android-package" value="nl.xservices.plugins.SocialSharing" />
</feature>
<!-- for Windows Phone -->
<feature name="SocialSharing">
  <param name="wp-package" value="SocialSharing"/>
</feature>
```

In config.xml file of the project.

Fourth, the plugin can be used in the PhoneGap build by adding

```
<gap:plugin name="cordova-plugin-x-socialsharing" source="npm" />
```

To the config.xml file [19] of the project. As the Ionic framework comes with support of Cordova CLI, the second option was used in this project.

After adding the plugin to the source of the project, it can be used as a dependency on the desired controller. In this case it was added to the post controller, which is bound to the post.html template.

The message for sharing is totally customizable. It can be a pre-defined message or it can be an empty message with a link. In both cases users can modify and change the message within a sharing link, so that they can have a personalized message to share in different social Medias.

```

68 // post controller.
69 .controller('postCtrl', function(
70   $scope,
71   $http,
72   $stateParams,
73   $sce,
74   $localStorage,
75   $cordovaSocialSharing) {
76   $http.get('http://allfashion.mobiproj.com/wp-json/wp/v2/posts/' + $stateParams.postId, {cache: true}).then(
77     function(returnedData){
78       $scope.postDetails = returnedData.data;
79       $scope.post_title = $sce.trustAsHtml($scope.postDetails.title.rendered);
80       $scope.post_content = $sce.trustAsHtml($scope.postDetails.content.rendered);
81       $scope.post_image = $scope.postDetails.better_featured_image.source_url;
82
83     }, function(err){
84       console.log(err);
85     })
86
87   $scope.share = function(title) {
88     //console.log(title);
89     $cordovaSocialSharing.share(title);
90   };
91
92 })

```

Figure 20. Using PhoneGap social media in controller

The plugin can be added as a dependency in a controller as it is illustrated in figure 20 on line 75. After that on line 76 posts are fetched according to their unique postId and their attributes are bound to the post_title, the post_content and the post_image. These attributes are shown in the post.html template. Later on line 87, a post can be shared according to its title with a customized message.

```

7 <ion-content overflow-scroll="true">
8   <div class="item-image" style="position:relative;" dir="rtl" ng-cloak>
9     
10    <p class="item-content item-text-wrap text-right p-text-decoration" ng-bind-html="post_title"></p>
11    <button class="item-font button icon icon-fw-share button-clear" style="color:#0a1612; font-size: 16px !important;"
12      ng-click="share(post_title + ' ' + 'از کافه بازار دانلود کن'"
13      dir="rtl">باز نظر</button>
14  </div>

```

Figure 21. Binding social sharing plugin to view template

The share button can be called with a share function in figure 20 line 87 and it can be tied to a view in figure 21 line 12. This function will get the post_title and also a customized message indicating that the user enjoys reading this post and wants to share it with others.

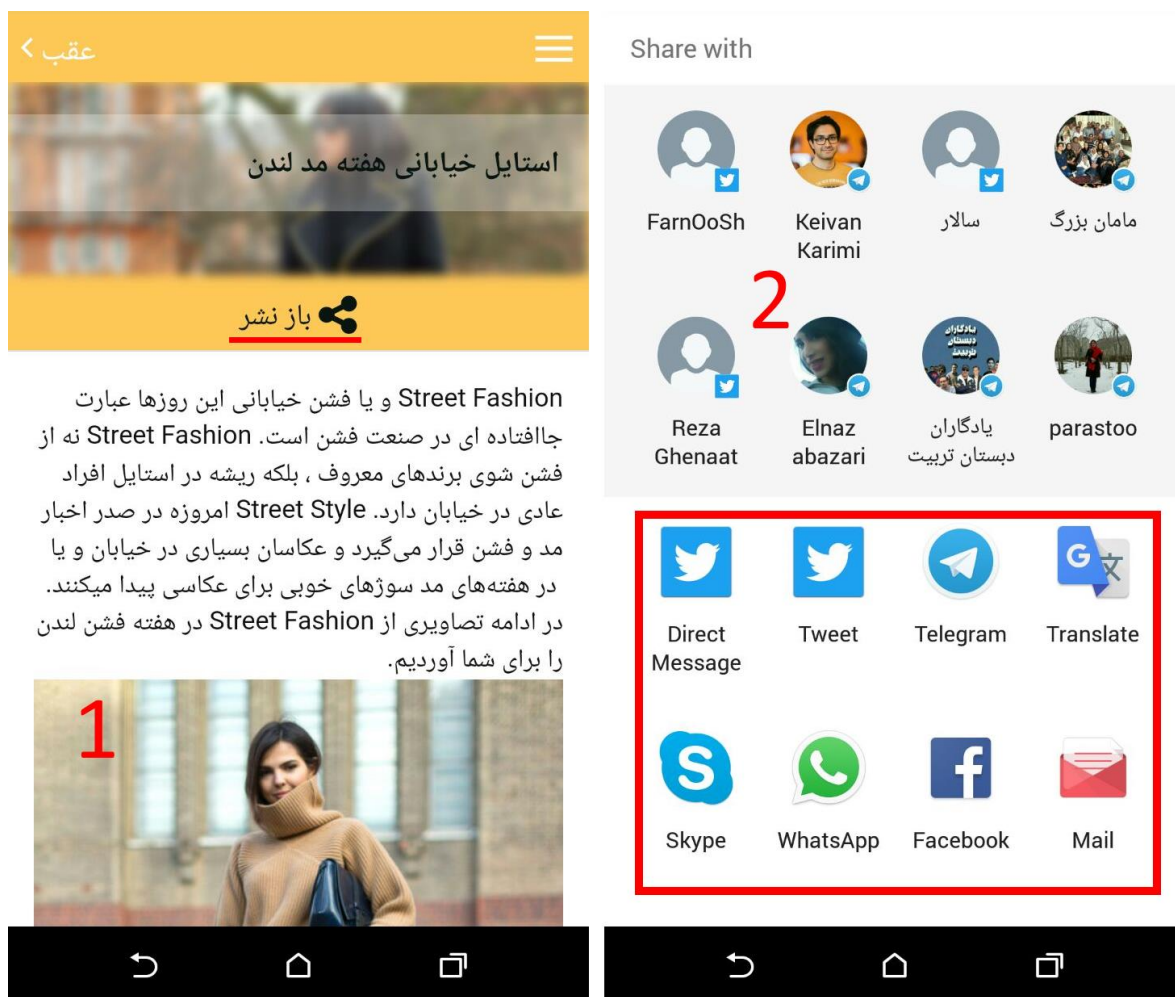


Figure 22. PhoneGap social sharing result in application view

In the final stage, the share functionality is useable by touching the share button icon in figure 22 part one. After touching the share button, right window which is the native sharing feature of the Android OS will be appeared. In this stage users can choose different applications that are available on their devices to share the post title with a predefined message in figure 21 line 12.

7.4 Installing and using the Ionic-toast plugin

Communicating with users inside application is an important matter. There are multiple methods to show notifications to users in different views of the application. Methods such as showing alerts, popups, modals or toasts. Alerts and popups are not common any-more and users feel irritated when see one of them. On the other hand, toasts are among the best solutions to show short messages.

The plugin can be installed by using this Bower command:

```
bower install ionic-toast --save
```

After that a specified path should navigate to the ionic-toast.bundle.min.js file in index.html file of the project:

```
<!-- path to ionic / angularjs files-->
<script src="lib/ionic-toast/dist/ionic-toast.bundle.min.js"></script>
```

```

1
2
3 angular.module('deepBlue', ['ionic', 'ionic-toast', 'deepBlue.controllers', 'deepBlue.services', 'ngStorage', 'ngCordova'])
4
5 .run(function($ionicPlatform, $rootScope, $timeout, $state, $cordovaPush) {
6   $ionicPlatform.ready(function() {
7
8   })
9
10

```

Figure 23. Injecting ionic-toast in the application module

Then the ionic-toast dependency should be injected in the application module as it is shown in figure 23. Finally the toast can be used as a function in the desired controller. [20]

The Ionic-toast has three useable styles. It can be used as top of the page toast, middle of the page toast or bottom of the page toast. There is also a possibility to customize the duration of appearance of a toast in addition to autohide feature. In this case the toast can be hidden after a specific time or it can be dismissed by users.

```

186 // controller for "app.catPosts" view
187 .controller('catPostsCtrl', function(
188     $scope,
189     $ionicListDelegate,
190     $http,
191     $sce,
192     $ionicScrollDelegate,
193     $localStorage,
194     ionicToast) {
195
196     $scope.doRefresh = function(){
197         $scope.recentPosts = [];
198
199         $http.get("http://allfashion.mobiproj.com/wp-json/wp/v2/posts?per_page=15").then(
200             function(returnedData){
201                 $scope.recentPosts = returnedData.data;
202                 //console.log($scope.recentPosts);
203                 $scope.recentPosts.forEach(function(element, index, array) {
204                     element.excerpt.rendered = element.excerpt.rendered.substr(0, 150);
205                     element.excerpt.rendered = $sce.trustAsHtml(element.excerpt.rendered);
206                     element.title.rendered = $sce.trustAsHtml(element.title.rendered);
207                     if($scope.Favorites.indexOf(element.id) != -1) {
208                         element.isFavorite = true;
209                     } else {
210                         element.isFavorite = false;
211                     }
212                 })
213                 ionicToast.show('دز حال مقامده آخرین مطالب مستید', 'top', false, 2500);
214             }, function(err){
215                 ionicToast.show('دز حال حاضر امکان به روز رسانی مطالب وجود ندارد', 'top', false, 2500);
216                 console.log(err);
217             })
218

```

Figure 24. Using ionic-toast in controller

As shown in figure 24, the ionicToast is injected as a dependency on line 194 and it can be used as a readymade function on lines 213 and 215.

On line 199 after sending a GET request and receiving data, a toast message will be received and shown to users regarding to success or failure in updating the GET request. In case of a success call back, users will get a success message. Which is shown on line 213. It will be shown on top of the application and it will be disappeared after 2.5 seconds. In case of happening any errors or unsuccessful call back, users will receive another message from line 216. The message indicates that there is a connection error or a problem and it will disappear after 2.5 seconds.

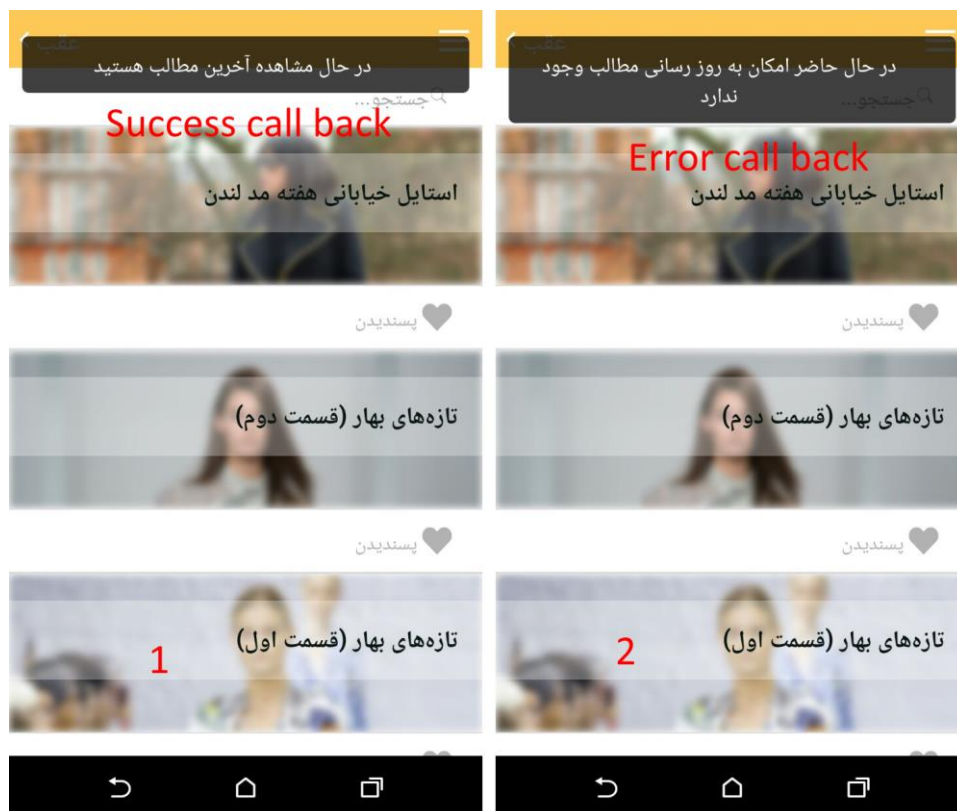


Figure 25. Ionic toast results in application view

The `ionicToast` function is nested inside the `$scope.dorefresh` function on line 196, so any time users try to refresh the `catPosts.html` template, they will receive desired messages. The view of the toasts in application is shown in figure 25.

8 Network security test

8.1 Man-in-the-middle packet sniffing

Man-in-the-middle or MiM is the process of re-routing user's network traffic through a new path so that a third party can sniff or modify packets of data.

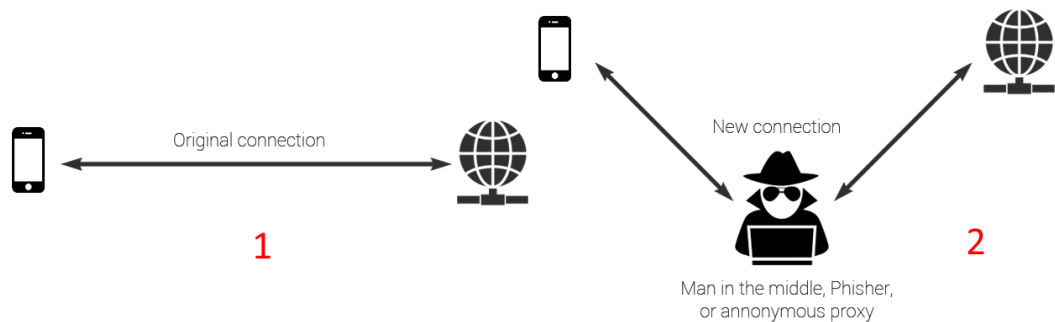


Figure 26. Man in the middle. Modified from [21]

As shown in figure 26 part one, the original connection is between a user and the Internet network. Imagine a shopping mall, where users may not have access to cellular data on their sim card. A MiM attacker can create an open Wi-Fi hotspot, so that users are encouraged to connect to this free connection to access the internet. This is the situation that happens in part two of figure 26. In this scenario MiM attacker can sniff and listen to any unencrypted traffics that are sent or received between users and the internet.

The process of packet sniffing can be done with different softwares. Wireshark is one of the most famous ones and it is free. As an experiment the Shiksho application was tested against the MiM packet sniffing.

8.2 Installing and using the Wireshark

Wireshark, which is one of the best tools for packet-level and network analysis, covers over 1000 protocols and is available as an open source solution [22, p.7]. By using Wireshark users can dynamically monitor traffic running on certain computers in the networks. It supports major operation systems such as Windows, OS X, Linux and UNIX. A wide range of users such as network professionals, security experts and developers can benefit of the Wireshark. [23]

Wireshark uses two different libraries to capture and filter packets. WinPcap for Windows and libpcap for Mac/Linux/Unix [22, p.7]. For the testing purpose of this project Windows 7 workstation was chosen.

The Wireshark software package can be downloaded from the Wireshark main website, <https://www.wireshark.org>, and the installation wizard will help users to install the software on their machines.

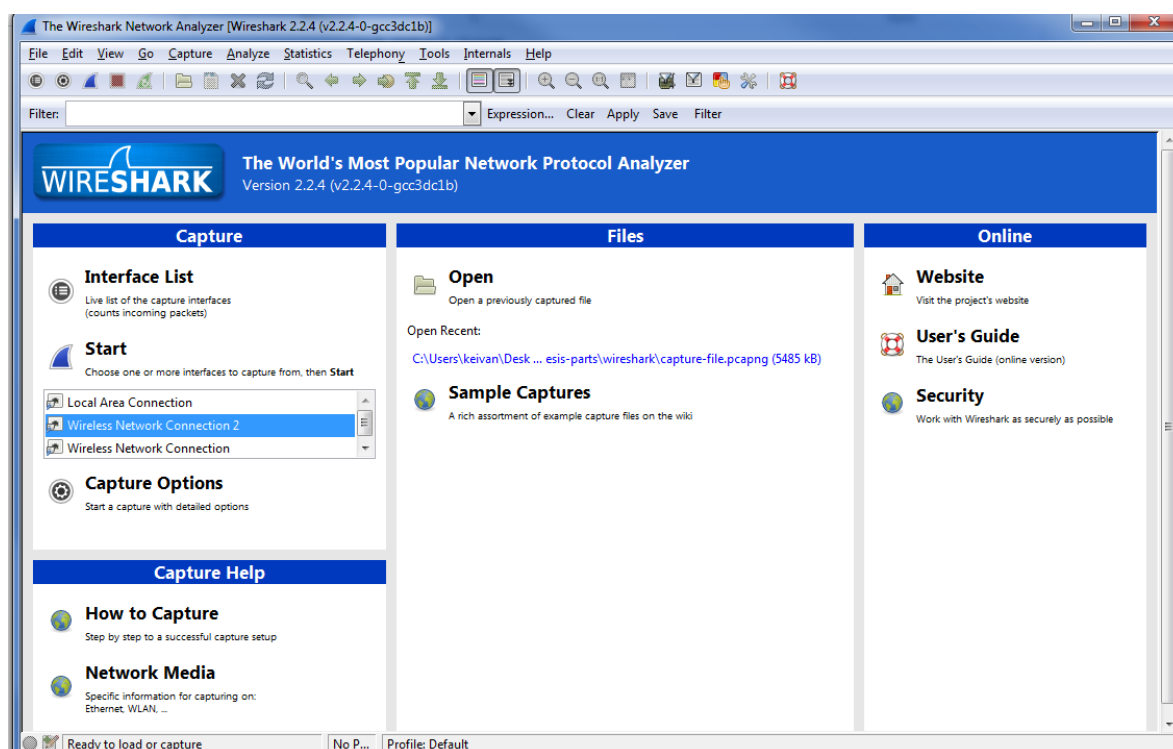


Figure 27. Wireshark first page

After installing and running the Wireshark software, the page on figure 27 will be displayed. In this panel users can access the available interfaces for monitor, check for previously captured files, ask for online help or use help documentations.

By selecting a target interface, in this case Local Area connection, and pressing the start, capturing and monitoring packets will be started. After capturing certain packets the process of capturing can be stopped by pressing the stop button. The captured packets can be stored in an external file, so that they will be available for further analysis.

8.3 Building a Wi-Fi hotspot and packet sniffing

For the experimental purpose of the Shiksho project, there was a need to build a Wi-Fi hotspot to have an isolated network. It is possible to monitor an isolated network using Wireshark. There are multiple ways to build a Wi-Fi hot spot. One way is to use a third-party software. In this case the Virtual Router software was hired.

Virtual Router, <https://virtualrouter.codeplex.com>, is a software to share a LAN/3g/4g network connection by using a Wi-Fi card. It is available under Ms-PL, Microsoft public license. In this experiment the free version was used.

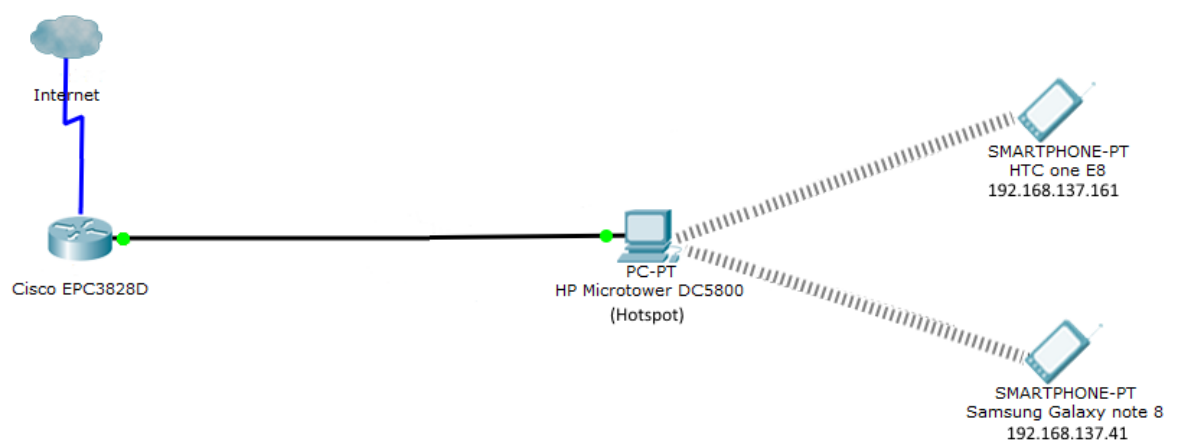
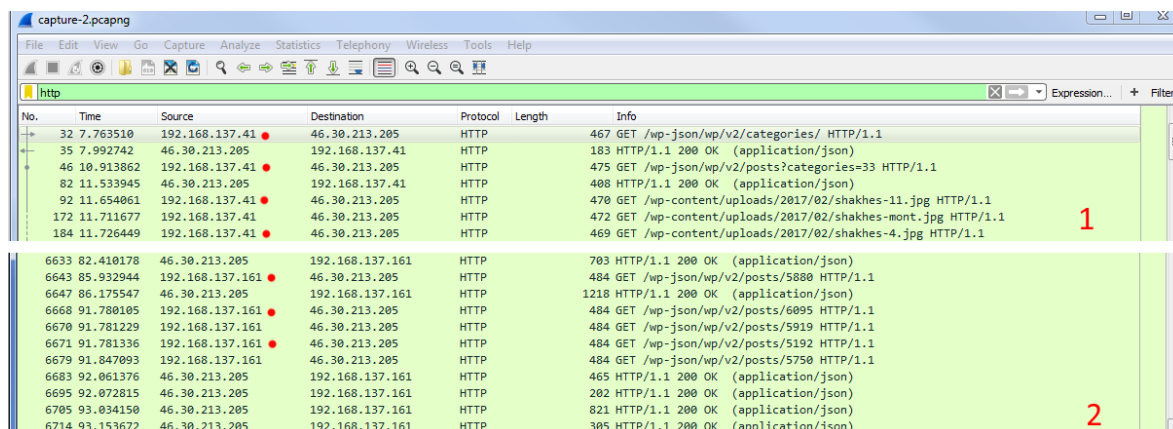


Figure 28. Network map

As illustrated in figure 28, the experiment network consist of multiple devices. The main Router is a Cisco EPC3828D with the default Cisco firmware. It provides internet access for the HP Microtower pc and this HP pc acts as a Wi-Fi hotspot for two smart devices. All the internet traffics for the HTC one e8 smart phone and the Samsung Galaxy note 8 tablet passes through the HP pc. With figure 28 scenario, it is possible to monitor network packets for the smart phone and the tablet by using Wireshark on the HP pc.

As stated in figure 27, there are multiple interfaces to monitor and capture packages from them. Since the HP microtower acts as a hotspot, the wireless network connection 2 was monitored. By capturing packets from this interface and simultaneously running the Shiksho application on the HTC one E8 and the Samsung galaxy note 8, different packets start showing on Wireshark. The packets are SSDP, MDNS, ARP, TCP, DHCP, and HTTP. The HTTP packets are the target in this packet sniffing process. The Wireshark offers a filtering system that users can filter the results according to their needs.

As a result, by using the filtering method only HTTP packets are visible. These HTTP packets contain all the \$http.get requests that the Shisko application sends and receives. In fact, these packets include the request and the response links and data.



No.	Time	Source	Destination	Protocol	Length	Info
32	7.763510	192.168.137.41	46.30.213.205	HTTP	467	GET /wp-json/wp/v2/categories/ HTTP/1.1
35	7.992742	46.30.213.205	192.168.137.41	HTTP	183	HTTP/1.1 200 OK (application/json)
46	10.913862	192.168.137.41	46.30.213.205	HTTP	475	GET /wp-json/wp/v2/posts?categories=33 HTTP/1.1
82	11.533945	46.30.213.205	192.168.137.41	HTTP	408	HTTP/1.1 200 OK (application/json)
92	11.654061	192.168.137.41	46.30.213.205	HTTP	470	GET /wp-content/uploads/2017/02/shakhes-11.jpg HTTP/1.1
172	11.711677	192.168.137.41	46.30.213.205	HTTP	472	GET /wp-content/uploads/2017/02/shakhes-mont.jpg HTTP/1.1
184	11.726449	192.168.137.41	46.30.213.205	HTTP	469	GET /wp-content/uploads/2017/02/shakhes-4.jpg HTTP/1.1
6633	82.410178	46.30.213.205	192.168.137.161	HTTP	703	HTTP/1.1 200 OK (application/json)
6643	85.932944	192.168.137.161	46.30.213.205	HTTP	484	GET /wp-json/wp/v2/posts/5880 HTTP/1.1
6647	86.175547	46.30.213.205	192.168.137.161	HTTP	1218	HTTP/1.1 200 OK (application/json)
6668	91.780105	192.168.137.161	46.30.213.205	HTTP	484	GET /wp-json/wp/v2/posts/6095 HTTP/1.1
6670	91.781229	192.168.137.161	46.30.213.205	HTTP	484	GET /wp-json/wp/v2/posts/5919 HTTP/1.1
6671	91.781336	192.168.137.161	46.30.213.205	HTTP	484	GET /wp-json/wp/v2/posts/5192 HTTP/1.1
6679	91.847093	192.168.137.161	46.30.213.205	HTTP	484	GET /wp-json/wp/v2/posts/5750 HTTP/1.1
6683	92.061376	46.30.213.205	192.168.137.161	HTTP	465	HTTP/1.1 200 OK (application/json)
6695	92.072815	46.30.213.205	192.168.137.161	HTTP	202	HTTP/1.1 200 OK (application/json)
6705	93.034150	46.30.213.205	192.168.137.161	HTTP	821	HTTP/1.1 200 OK (application/json)
6714	93.153672	46.30.213.205	192.168.137.161	HTTP	305	HTTP/1.1 200 OK (application/json)

Figure 29. MiM packet monitoring with Wireshark

As illustrated in figure 29, in the part one of the image there are multiple GET requests from 192.168.137.41, which is the Samsung Galaxy note 8. On request number 32, the device is trying to access categories of the application from the backend. This is the same as the GET request on line 45 in figure 30.

```
45 $http.get('http://allfashion.mobiproj.com/wp-json/wp/v2/categories/', {cache: true}).then(
```

Figure 30. \$http.get request for categories

Latter on the device is trying to access images of different categories on the packet 92, 172 and 184. By inspecting these kind of packets as MiM, the URL to source image are accessible. As an example in figure 31, packet 92 was inspected. The GET request for an image file is clearly visible under the Hypertext Transfer Protocol, HTTP. This URL is the internal URL request and the path is /wp-content/uploads/2017/02/shakhes-11.jpg. There are other fields that reveal more information. Under the host, the main URL of website that hosts the back-end is visible.

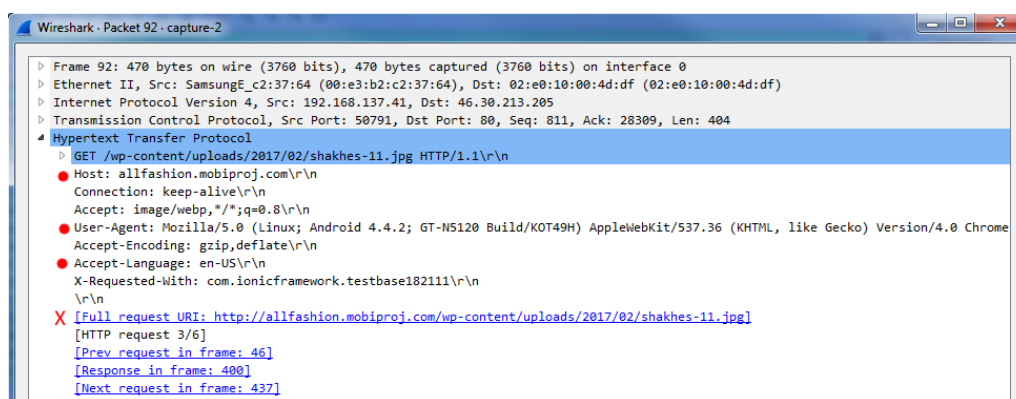


Figure 31. MiM Inspecting a single image request packet

Under the user-agent, information about the Android version of the requesting device is visible, which is 4.4.2. On the accept-Language field, the main language of the device, Galaxy note 8, is declared as en-US. Finally, the full URL to desired image is visible next to the red X mark. By clicking on this URL, the image that device is trying to access is visible to the packet sniffer and MiM.

The same data is visible for the second device, the HTC E8 with the IP address of 192.168.137.161. As it is shown in figure 29 part two, multiple packets such as 6643, 6670 and 6671 are trying to access different posts.

```
$http.get('http://allfashion.mobiproj.com/wp-json/wp/v2/posts/' + $stateParams.postId, {cache: true}).then(
```

Figure 32. \$http.get request for posts

These packets are the result of the Get request for posts as shown in figure 32.

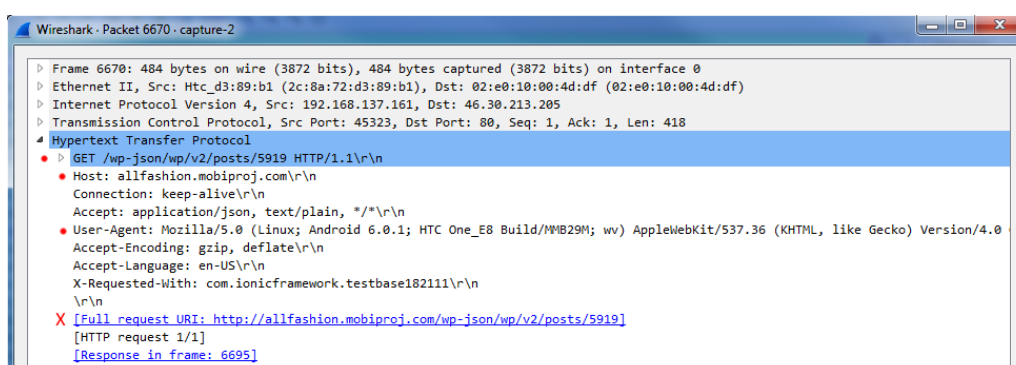


Figure 33. MiM inspecting a single post request packet

In figure 33, the information about requests from second device, HTC one E8, is visible. The user agent is Android 6.0.1 and under the host information about back-end and the hosting website is visible. This time the request is for a specific post and the internal URL is /wp-json/wp/v2/posts/5919, which is generated by \$http.get request from figure 32. At last the full URL to that specific post is visible next to the red X mark In figure 33.

The experiment shows that http requests are not secure enough to send or receive sensitive data. In other words, if someone tries to sniff packets that are being transferred between the Shiksho application and the back-end site, he/she could easily do it.

There are multiple ways to secure this data transmission such as using secure \$services on AngularJS or hosting the back-end part of the application on https service, but they are not within the scope of this project.

9 Conclusion

In conclusion, the target of this project was to build a mobile application using available tools and technologies. As a result, the latest available technologies were hired to set up a development environment and to develop and build the application. Finally an application was delivered to users, so that they can follow and read the latest news about fashion.

The application was deployed as a hybrid application, which is different from the common native applications in the Android and the iOS ecosystem. A native application tends to perform smoother and have better performance on mobile devices. However, during the past couple of years hybrid applications and their deployment methods have been improved. Nowadays, hybrid applications perform as well as native applications and their performances are trustworthy. The Shiksho application was tested on multiple states from end-to-end tests to security tests and it turned out it has acceptable quality.

Another aspect worth mentioning was the workflow. At the beginning of the project, the project was anticipated to be completed in two weeks rather than a month. Even though I had enough knowledge about web development, I experienced some difficulties in the development and deployment areas. As a result, I did some more studies on data structure and the foundation of hybrid application development. Alongside I did hours of searching for fixing minor bugs and issues with the framework, plugins and even the development OS itself.

The Shiksho application is under constant development of adding new features. It is anticipated to roll out a new update for the application every two months and improve the user experience with each update. As the application was based on version one of the AngularJS framework, the development will be continued on this framework. In addition, there is a plan to update the application with the latest upcoming technologies and replace the structure with new syntaxes. The next major update would be deploying the Shiksho application using version two of AngularJS.

Deploying the application based on version two of AngularJS needs knowledge on TypeScript. Furthermore, the structure and implementation is different on the AngularJS version two than on version one.

References

- 1 Uzayr Sufyan bin (July 2016) Learning WordPress REST API: A practical tutorial to get you up and running with the revolutionary WordPress REST API, Birmingham, UK, Packet Publishing Ltd.
- 2 Beal Vangie, API - application program interface [online]. URL: <http://www.webopedia.com/TERM/A/API.html>. Accessed 5 March 2017.
- 3 Rouse Margaret (2016) DEFINITION RESTful API [online]. December 2016 URL: <http://searchcloudstorage.techtarget.com/definition/RESTful-API>. Accessed 5 March 2017.
- 4 Ende93 (2016), JSON [online]. URL: https://developer.mozilla.org/enUS/docs/Web/JavaScript/Reference/Global_Objects/JSON. Accessed 5 March 2017.
- 5 K Barry Douglas, Service Architecture Representational State Transfer (REST) [online]. URL: http://www.service-architecture.com/articles/webservices/representational_state_transfer_rest.html. Accessed 5 March 2017.
- 6 Mehta Bhakti (September 2014), RESTful Java Patterns and Best Practices: Learn best practices to efficiently build scalable, reliable, and maintainable high performance RESTful services, Birmingham, UK, Packet Publishing Ltd.
- 7 Postman Developing APIs is hard. Postman makes it easy [online]. URL: <https://www.getpostman.com/>. Accessed 5 March 2017.
- 8 WP REST API Schema [online]. URL: <http://v2.wp-api.org/reference/media/>. Accessed 5 March 2017.
- 9 Ravulavaru Arvind (July 2015) Learning Ionic: Build real-time and hybrid mobile applications with Ionic, Birmingham, UK, Packet Publishing Ltd.
- 10 Welcome to Ionic, How to get the most out of Ionic [online]. URL: <http://ionicframework.com/docs/guide/preface.html>. Accessed 5 March 2017.
- 11 Android Studio, The Official IDE for Android [online]. URL: <https://developer.android.com/studio/index.html#tos-header>. Accessed 5 March 2017.
- 12 Architectural overview of Cordova platform - Apache Cordova [online]. URL: <https://cordova.apache.org/docs/en/latest/guide/overview/index.html>. Accessed 5 March 2017.
- 13 Panda Sandeep (2014) AngularJS: Novice to Ninja. Victoria, Australia, SitePoint Pty. Ltd.
- 14 Branas Rodrigo (August 2014) AngularJS Essentials, Design and construct reusable, maintainable, and modular web applications with AngularJS. Birmingham, UK, Packet Publishing Ltd.

- 15 Ruebbelke Lukas (2015) AngularJS in action. Shelter Island, New York, United States of America, Manning Publications Co.
- 16 Shora Andy, Promises in AngularJS, Explained as a Cartoon, An alternative frontend development article [online]. URL: <http://andyshora.com/promises-angularjs-explained-as-cartoon.html>. Accessed 5 March 2017.
- 17 ngCordova - Document and Examples - by the Ionic Framework Team [online]. URL: <http://ngcordova.com/docs/>. Accessed 5 March 2017.
- 18 Egilkh, gsklee/ngStorage: localStorage and sessionStorage done right for AngularJS [online]. URL: <https://github.com/gsklee/ngStorage>. Accessed 5 March 2017.
- 19 Verbruggen Eddy, EddyVerbruggen/SocialSharing-PhoneGap-Plugin: 📱❤️📄🔗📱 Cordova plugin to share text, a file (image/PDF/..), or a URL (or all three) via the native sharing widget [online]. URL: <https://github.com/EddyVerbruggen/SocialSharing-PhoneGap-Plugin>. Accessed 5 March 2017.
- 20 Rajeshwarpatlolla, rajeshwarpatlolla/ionic-toast: 'ionic-toast' bower component for ionic framework applications [online]. URL: <https://github.com/rajeshwarpatlolla/ionic-toast>. Accessed 5 March 2017.
- 21 What is Man in the Middle Attack? | MITM Prevention Techniques [online]. URL: <https://securebox.comodo.com/ssl-sniffing/man-in-the-middle-attack/>. Accessed 5 March 2017.
- 22 Baxter James. H (October 2014) Wireshark Essentials, Get up and running with Wireshark to analyze network packets and protocols effectively. Birmingham, UK, Packet Publishing Ltd.
- 23 Wireshark - Frequently Asked Questions [online]. URL: <https://www.wireshark.org/faq.html#q1.1>. Accessed 5 March 2017.
- 24 Using HTTP Methods for RESTful Services [online]. URL: <http://www.restapitutorial.com/lessons/httpmethods.html>. Accessed 1 April 2017.
- 25 Dependency Injection [online]. URL: <https://docs.angularjs.org/guide/di>. Accessed 1 April 2017.

